单态（Singleton)

目录

单态模式的定义

- Singleton模式主要作用是保证在Java应用程序中，一个类Class只有一个实例存在。
- 在项目的很多地方都会用到它，比如说数据库的链接。
- 使用Singleton的好处还在于可以节省内存，因为它限制了实例的个数，有利于Java垃圾回收（garbage collection）。

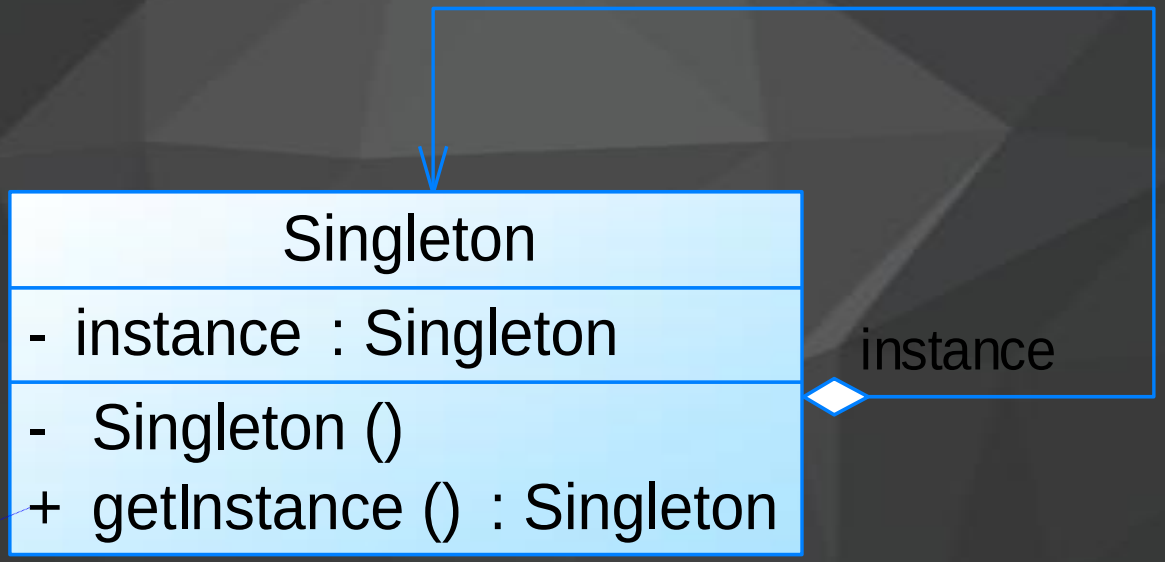
实现思路：

- 要想控制一个类只被创建一个实例，那么首要的问题就是要把创建实例的权限收回来，让类自身来负责自己类实例的创建工作，然后由这个类来提供外部可以访问这个类实例的方法，这就是单例模式的实现方式

实现方式：

- 确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例，这个类称为单例类，并提供一个访问它的全局访问点
- 三个要点：一是某个类只能有一个实例；二是它必须自行创建这个实例；三是它必须自行向整个系统提供这个实例

单态模式UML




```
if(instance==null)
        instance=new Singleton();
return instance;
```

单态模式分析

- 单例模式的目的是保证一个类仅有一个实例，并提供一个访问它的全局访问点。
- 单例模式包含的角色只有一个，就是单例类Singleton。
- 单例类拥有一个私有构造函数，确保用户无法通过new关键字直接实例化它。
- 包含一个静态私有成员变量与静态公有的工厂方法。

单态模式的几种实现

- 按照创建的时机不同分为:
  - 饿汉式

```
public class HungrySingle{
        //静态私有成员变量
    private static HungrySingle single = new HungrySingle();
    private HungrySingle(){} //私有构造函数
    //静态公有工厂方法，返回唯一实例
    public static HungrySingle getInstance(){
     return single;
     }
}
```

注意：由于构造器是私有的,因此此类不能被继承

单态模式的几种实现

```
public class LazySingle{
  private static LazySingle single = null;
  private LazySingle(){}

synchronized  public static  LazySingle getInstance(){
        if(single==null){
        single = new LazySingle();
        }
        return single;
  }
  }
```

这个属性就是用来缓存实例的

这里就体现了延迟加载，马上就要使用这个实例了，还不知道有没有呢，所以判断一下，如果没有，没办法了，赶紧创建一个吧~

懒汉式在第一次被引用时将自己实例化

由于构造器是私有的,因此此类不能被继承

创建实例涉及到资源初始化，最好使用同步机制

两种单态模式的区别

- **时间和空间**
  - 懒汉式是典型的时间换空间，也就是每次获取实例都会进行判断，看是否需要创建实例。如果一直没有人使用的话，那就不会创建实例，节约内存空间。（延迟加载: 开始不加载资源或者数据，等到马上就要使用这个资源或数据了才加载，所以也称Lazy Load(延迟加载)）
  - 饿汉式是典型的空间换时间，当类装载的时候就会创建类实例，不管你用不用，先创建出来，然后每次调用的时候，就不需要再判断，节省运行时间。

两种单态模式的区别

- **线程安全**
  - 从线程安全性上讲，不加同步的懒汉式是线程不安全的
  - 饿汉式是线程安全的，因为虚拟机保证了只会装载一次，在装载类的时候是不会发生并发的。

单态模式总结

- 单例类的构造函数为私有
- 提供一个自身的静态私有成员变量
- 提供一个公有的静态工厂方法

翡翠教育
EMERALD EDUCATION

谢　谢